

甄景贤 (King-Yin Yan)  
General.Intelligence@Gmail.com

逻辑 AI 那边，「结构」是高度的符号化抽象，但**学习算法太慢**；我的目的是建立一道「桥」，将逻辑 AI 的某部分结构**转移**到神经网络那边。

$$\begin{array}{ccc}
 F = \text{logic system} & & F = \text{recurrent deep NN} \\
 \text{---} & \approx & \text{---} \\
 x \cup \text{KB} \models x' & & x \text{ --- } x' \\
 x = \text{model} & & x = \text{mental state}
 \end{array}
 \quad (1)$$

## 1 神经网络的结构

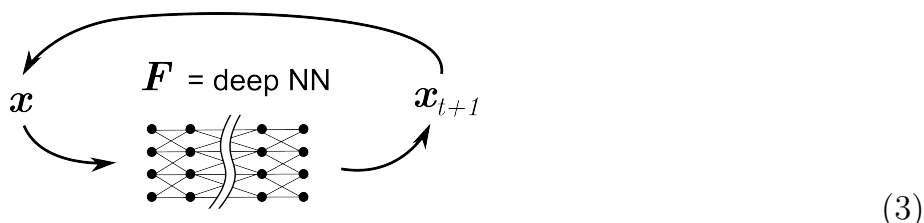
每层的权重矩阵      总层数

$$F(\mathbf{x}) = \mathcal{O}(W_1 \mathcal{O}(W_2 \dots \mathcal{O}(W_L \mathbf{x}))) \quad (2)$$

① 作用在  $\mathbf{x}$  的每个分量上，它的作用在座标变换下没有不变性。所以 ① 不是一个向量运算，从而  $\mathbb{X} \ni \mathbf{x}$  的结构也不是向量空间的结构。通常习惯把  $\vec{x}$  写成向量形式，但这有点误导。

如果将神经网络首尾相接造成迴路，这是一种智能系统的最简单形式。举例来说，如果要识别「白猫追黑猫」的视像，「猫」这物体要被识别两次，很明显我们不应浪费两个神经网络。

络去做这件事，所以 迴路是必须的：



它的状态方程是：

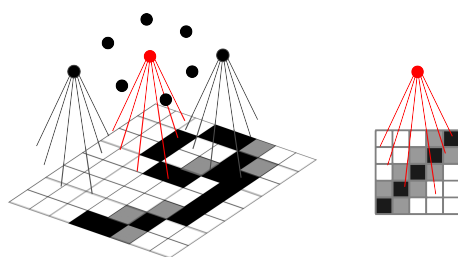
$$\boxed{\text{离散时间}} \quad x_{t+1} = F(x_t) \quad (4)$$

$$\boxed{\text{连续时间}} \quad \dot{x} = f(x) \quad (5)$$

由此可以看出， $\mathbb{X}$  是一个微分流形。更深入地讲，它是一个力学上的 Hamiltonian 系统，具有 symplectic（辛流形）结构。但这超出了本文范围，详见本篇的前篇 [16]。

现在思考一下，神经网络怎样识别模式，或许会有帮助....

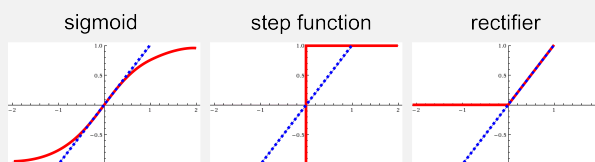
考虑最简单的情况，例如提取数字“9”的特徵的一层网络。这层网络可以有很多神经元（左图），每个神经元局部地覆盖输入层，即所谓视觉神经元的 local receptive field（右图）。



假设红色的神经元专门负责辨识「对角线」这一特徵。它的方程式是  $y = \mathcal{S}(Wx)$ 。矩阵  $W$  的作用是 affine「旋转」特徵空间，令我们想要的特徵指向某一方向。然后再用  $\mathcal{S}$ 「挤压」想要的特徵和不想要的特徵。Sigmoid 之后的输出，代表某类特徵的存在与否，即  $\{0, 1\}$ 。这是一种资讯的压缩。

叉开话题，讲一点 chaos theory:

$\mathcal{S}^{-1}$  的作用是「扯」(stretch)，将本来邻近的两点的距离非线性地拉远。看看以下各种常见的激活函数，它们全都是相对於 identity  $y = x$  的非线性 deformation:



这和 Steven Smale 提出的「马蹄」[12] 非常类似，它是制造混沌的处方之一。Smale 马蹄的另一个变种叫做 baker map，其作用类似於「搓面粉」。换句话说，「拉扯」然后放回原空间，如此不断重复，就会产生混沌 [3] [13]。

这里有一点重大意义： $F^{-1}$  有混沌的典型特徵：它「对初始状态的微少变化非常敏感」。根据我的观点，神经网络的正向运作是 pattern recognition = 资讯压缩，反向是反压缩。反向的时候，因为同一概念可以对应於很多不同的输入 ( $F^{-1}(x)$  可以是很多 patterns)，所以输出的微少变化（例如 0.99 和 0.98）会造成  $F^{-1}$  的极大起伏；换句

话说即是有混沌现象。换句话说  $F$  的逆是 unpredictable 的，简言之，神经网络  $F$  是不可逆的压缩过程。

最近一个有趣的例子是 DeepDream [1]，它用神经网络的  $F^{-1}$  产生有迷幻感觉的 pre-image，证实了  $F^{-1}$  可以和原本的 image 差距很大：



(8)

在神经网络的正向运作时似乎没有这种 “stretching”，这似乎不会产生混沌，而且根据 contraction mapping theorem,  $F$  的 iteration 会终止於 fixed point(s)，但前提是  $F$  是 contractive 的，换句话说 the spectral radius of the Jacobian matrix of  $F \leq 1$ ，但这一点我暂时未能确定（如果对  $W$  没有限制，这似乎不成立）。

另方面，如果反向是混沌，正向似乎也应该是混沌；在 ergodic theory 里可以计算动态系统的 topological entropy，而这个 entropy 的值是 time-reversal invariant 的。我暂时不清楚如果用别的 entropy 计算会不会不同，有待请教一下 ergodic theory 方面的人....？

总括以上，可以归结出一个原则：

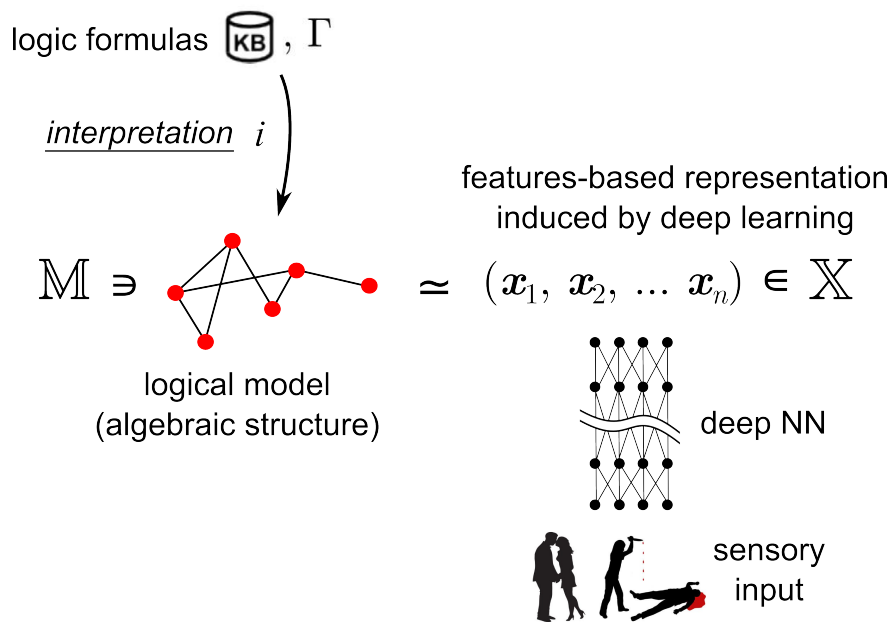
- 每个神经元的输出代表某个 feature 的存在与否
  - 更高层的神经元代表下层 features 之间的关系
- (9)

这些原则暂时未能严格地表述和证明，或者可以叫它做 **神经原理 (Neural Postulate)**。

凭这个思路推广，可以推测这样的 correspondence:

$$\begin{array}{llll}
 \text{M} & \simeq & \text{X} & \\
 \text{逻辑物体} & \bullet & \Leftrightarrow & \text{neuron} \\
 \text{逻辑关系} & \bullet\text{---}\bullet & \Leftrightarrow & \text{relation between higher and lower neurons}
 \end{array}
 \tag{10}$$

从 model theory 的角度可以这样理解：



## 2 逻辑的结构

一个逻辑系统可以这样定义：

- 一些 constant symbols, predicate symbols, 和 function symbols
- 由上述的原子建立 **命题** (propositions)
- 命题之间可以有连接词： $\neg, \wedge, \vee$  等
- 建立 **逻辑后果** (consequence) 关系： $\Gamma \vdash \Delta$

逻辑 AI 的 learning algorithm 叫 ILP (inductive logic programming), 这已经是一个 well-established field, 如有兴趣可参看我的 ILP tutorial (这版本有点旧, 有待更新) [14] 或 de Raedt 的教科书 [10]。ILP 在逻辑式子的符号空间中进行 combinatorial search, 缺点是太慢。神经网络的 gradient descent (ie, back-propagation) 快很多, 暂时未知是不是要混合这两种算法, 还是只用 back-prop....?

我们的目的是将逻辑的结构 transfer 到神经网络。逻辑的结构有两方面：

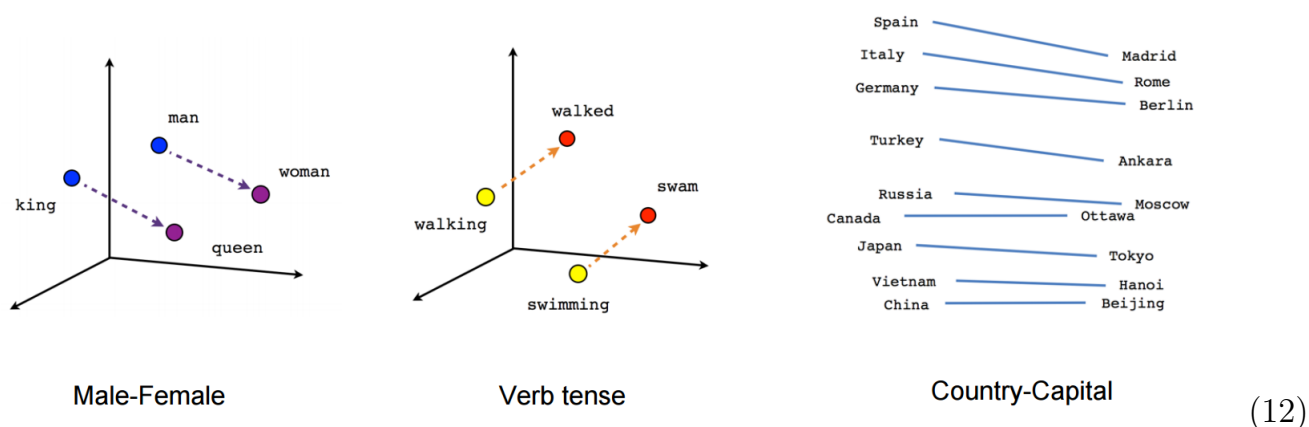
- 命题的内部结构 (sub-propositional structure)
- 命题之间的结构 (= mental state 的结构)

### 2.1 命题内部结构

谓词逻辑的命题结构比较特别, 所以我初时专注於分析它。但后来发现其实 **命题-level** 的结构 (§2.2) 才是比较重要的 (即它更能缩小搜寻空间)。

## 2.1.1 Semantic distance 与 compositionality

我们的目的是想将整套逻辑 AI 的器材搬到 **连续**的微分流形上去实现。这个做法，是受了 Google 的 Word2Vec [9] 算法的启发，它可以将自然语言的词语 embed 到度量空间中，而且词语之间的距离是 **semantic distance**（语义学上的距离）：



当然，word2vec 一经发表之后，很多人开始构思怎样把「句子」也 embed 到度量空间上。第一个想法当然是用 **tensor product**，例如 “I love you” 这句话就变成了  $I \otimes \text{love} \otimes \text{you}$ 。

但这个做法表面上很好，但细思之下却有问题。在语言学 / 逻辑学里有一个重要概念叫 **compositionality**，它说：

合成物的语义 (semantics) 由其所构成的原子生成，而不依赖于其他资料                      (13)

例如「我爱你」由「我、爱、你」三个原子组成。

但以下是一个反例：

- John 对 Jennifer 说「我爱妳」
- John 对 Jessica 说「我爱妳」
- Jenifer 发怒走了

「妳」字的意思在第 1、2 句中分别指 Jennifer 和 Jessica，如果简单地用「乘积」乘出来，欠缺了 pronoun resolution，就会导致谬误。这牵涉到每个词语在句子中的 interpretation \*。所以句子的意义 不是简单地由词语 bottom-up 生成的。

用 tensor product 的做法，得出的空间结构是 syntactic 而不是 semantic 的，但神经网络学习的重点是 **泛化** (generalization)，它基於「邻近的点的意义相近」才能成功，所以一定要 semantic distance。

\* 例如代名词、象徵和比喻等。经典 AI 研究者 Jerry Hobbs 开发了自然语言智能系统 Tacitus [5] [4]（这个字有「沉默」的意思），他的理论是基於逻辑上的 abductive interpretations（即逻辑上反方向推导，寻找句子意义的解释），没有这种解释的话，将句子分拆成词语 并不足以组成句子的完整意义；亦即是说，naïve compositionality 在人类语言中并不成立。

### 2.1.2 基於 “features” 的逻辑

上节看到 tensor product 的做法似乎有问题，它用原子 “algebraically” 组合成句子，原子之间的距离可以是语义学相近的，但「乘出来」的句子在语义空间上的位置可能 **很不准确**。另一方面，在神经网络里，「状态」 $\mathbf{x}$  是由一组 features 组成的，换句话说，它是一些 features 的 conjunction（也可以看成是乘积）。两者都是乘积，但前者有问题，因为它是由自然语言的字 / 词构成的，而后者是由概念原子 (conceptual atoms / features) 构成的。所以，如果我们避免将自然语言的字 / 句 naïve 地映照到 representation 上，使用概念原子的乘积应该没有问题。

暂时只考虑一个 thought (= 命题) 如何表示。

假设思维空间的 metric 是语义的（语义相近则点的距离相近）。根据「神经原理」(9)，每个 thought 必然是由一组 features 表述，形如  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ ，每个  $x_i$  是一个 feature 或概念原子。例如：

$$\mathbf{x} = \text{我很肚饿} = \text{关于我的} \cap \text{关于生理的} \cap \text{关于食物的} \cap \text{负面的} \cap \dots \quad (14)$$

这些 features 可以数量很多，它们是 **sub-symbolic** 的，很多 features 的乘积构成通常意义下的 symbols。

总括来说：一个 thought 是一组 binary features 的乘积。

### 2.1.3 谓词逻辑中的 linkage 现象

在经典谓词逻辑 (predicate logic) 中有一个 “linkage” 的现象，例如以下这个式子（爸爸的爸爸是爷爷）：

$$\forall X \forall Y \forall Z. \text{grandfather}(\mathbf{X}, \mathbf{Z}) \leftarrow \text{father}(\mathbf{X}, \mathbf{Y}) \wedge \text{father}(\mathbf{Y}, \mathbf{Z}) \quad (15)$$

那意思是说，无论右边的变量（例如 X）怎样改变，左边的 X 必须代入相同的值。这是代入 (substitution) 的本质。

我们想将 predicate logic 的结构「搬到」神经网络，从抽象意义上说可以有很多种做法，现我讲述一种我设计的比较简单直接的做法：

状态空间的 transition  $\mathbf{x} \mapsto \mathbf{y}$  看成是逻辑推导  $\mathbf{x} \vdash \mathbf{y}$ ，那么 linkage 就是  $\mathbf{x}$  的  $i$ -分量到  $\mathbf{y}$  的  $j$ -分量的 identity map。换句话说，有形如下式的一些 id maps 「埋藏」在  $\mathbf{F}$  里面：

$$\mathbf{F} : (x_1, \dots, \mathbf{x}_i, \dots, x_n) \mapsto (y_1, \dots, \mathbf{y}_j, \dots, y_n) \quad (16)$$

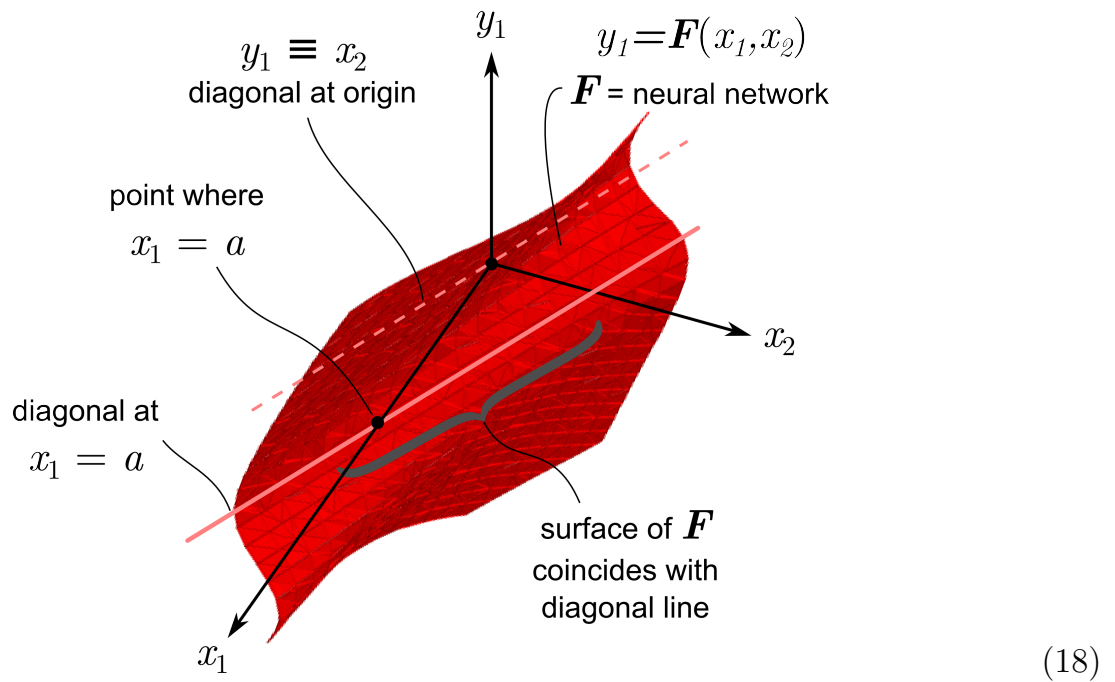


注意，我说「埋藏」的意思是：这些 linkages 只在  $\mathbf{x}$  的某些位置才存在（例如，当前提中出现了两个关于「爸爸」的命题时）。更准确地说：

$$\mathbf{F} : \begin{cases} y_j \equiv x_i, & \text{if } \mathbf{x} = \hat{\mathbf{a}} \text{ for some coordinates except } x_i \\ y_h \equiv x_k, & \text{if } \mathbf{x} = \hat{\mathbf{b}} \text{ for some coordinates except } x_k \\ \dots \text{ etc } \dots \\ \text{is free otherwise} \end{cases} \quad (17)$$

以上每条等式代表一个 linkage。

一个 linkage 的几何图像如下：

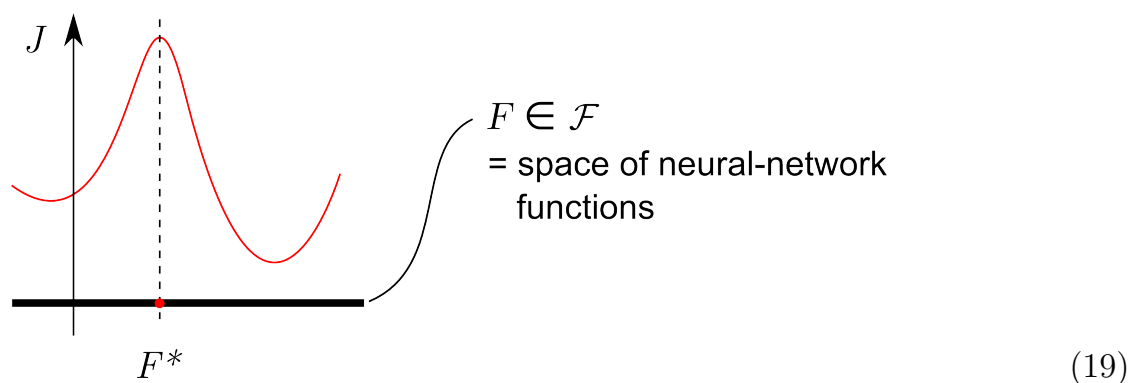


实际应用中  $\dim \mathbf{x}$  可能达到成千上万，其中可以有很多 linkages，很难 visualize。

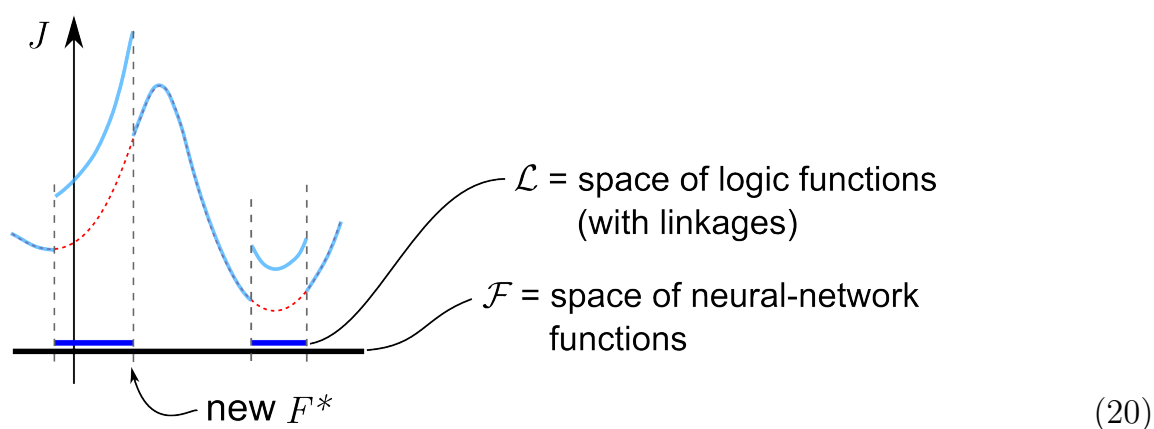
或者稍为形象化地解释一下：generalization 是智能的一个重要方面，例如由「苏格拉底会死、柏拉图会死」generalize 到「所有人都会死」这样的规律。这种泛化的 **propensity** 是一把双刃剑：有时候可以很聪明，但有时候过度泛化。例如有些男人被女性欺骗后，从此觉得所有女人都说谎。神经网络是一个空间中的 smooth function，它泛化的方法是靠 smoothness：某点的函数值改变时，该点的邻域的函数值也随著类似方向改变。神经网络本身没有那种 diagonal 式的泛化倾向（那是逻辑的特徵），但可以模拟它。神经网络  $\mathbf{F}$  对应於逻辑中的  $\vdash$ ，即是由一些前题推出结论； $\mathbf{F}$  是一个非常高维数的空间上的一个 hyper-surface。 $\mathbf{F}$  里面包含著各种 logic rules，例如在「人」和「死」这些位置上埋藏了「所有人都会死」这个 diagonal；而在「爸爸」和「爷爷」之间埋藏了「爸爸的爸爸是爷爷」这个 diagonal。 $\mathbf{F}$  的函数曲面必须经过这些 diagonals。如果没有逻辑的约束， $\mathbf{F}$  的函数曲面就是自由的 smooth function 了。

Now，我们关心的是怎样“combine”逻辑结构和神经网络结构。在这里我再选择一个特别简单的方法：在 (17) 中的那些等式，构成了一些约束在神经网络的  $W$  之上的 equational constraints。只需要用这些 constraints 去训练神经网络，就可以实现 linkages。

传统的 back-prop 搜寻  $F$  的最优值  $F^*$ ,  $J$  是 score-function。以下是一个示意图, 注意底下的 domain 是 泛函空间 (function space):

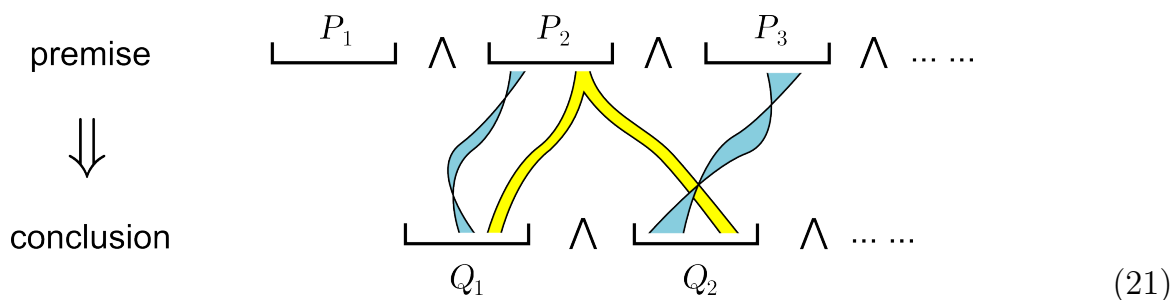


而我们需要的是新的 back-prop, 它在「混合」的 function space 上 optimize (至於 exactly 怎样混合我还不清楚),  $J$  在 logic 的 sub-space 上分数较高:



根据神经网络的 **universal approximation theorem** [2],  $L > 1$  层神经网络的函数  $F$  在函数空间中是 dense 的。换句话说, 如果神经元的个数充分多, 必然可以学习有任意 linkages 的函数。\*\*

## 2.1.4 有 linkage 的 learning algorithm



\*\* 试试实际上 solve 一个 constraint。为简单起见, 弃用  $\odot$  而用  $\ominus$  (反正实践中很多深度网络都用 rectifier), 这样很方便, 因为 rectifier 的前半截就是直线的 identity function。粗略地看, 如果只有一层, 那 constraint 约束了矩阵  $W$  的其中一行 (有  $n$  个分量); 每个 linkage 使用 2 个自由度 ( $\equiv$  等式使用 1 度, if-then 使用 1 度)。当层数增加时, 涉及到的  $W$  数目递增, 而约束等式仍然只是一条, 换句话说, 应该有很多的自由度可以用。这情况是乐观的。



```

Data: training data?
Result: learn linkages
initialization;
while not at end of this document do
  read current;
  if identity seems to exist then
    learn the identity;
    while interleaving with dual-brain normalization;
  else
    find next identity;
  end
end

```

### 2.1.5 用 relation algebra 如何?

以前曾经对 relation algebra [11] [8] 有些憧憬，因为它比较接近人类自然语言，但其实 relation algebra (RA) 和 first-order logic (FOL) 基本上是等效的，在 FOL 里面有 linkage 的复杂性，但在 RA 里面这个复杂性其实也没有消失。可以说「复杂度是守恒的」\*\*\*。

举例来说，在 (15) 中表达「爸爸的爸爸是爷爷」，可以用 RA 更简单地表达：

$$\text{father} \circ \text{father} = \text{grandfather} \quad (22)$$

实际的推导是这样的：

$$\begin{array}{l} \text{John father Pete} \\ \text{Pete father Paul} \\ \text{John father} \circ \text{father Paul} \end{array} \quad (23)$$

这时要 代入 上面的等式才能得出结论：

$$\text{John grandfather Paul} \quad (24)$$

所以 linkage 的复杂性变成了代数 formula 长度的复杂性。

### 2.1.6 Fractal 结构?

「长度」本身没有不妥，但我们的目的是将逻辑式子嵌入到状态空间  $x$  里，这时 variable length 令人很头痛，但 fixed length 没有此问题。如果硬要将 variable length 的式子嵌入到（有限维）向量空间中，似乎必须用到 **fractal** 结构（因为它有自相似性），同时需要设计一种新的神经网络，它先天性地有 fractal 结构在里面；但这比较复杂，我没有在这方向 explore。

\*\*\* 这句话是 category theorist Eugenia Cheng 说的。

## 2.1.7 命题结构 summary

逻辑命题的**内部** (sub-propositional) 结构，传统上有两种方法表达：一是 predicate logic，二是 relation algebra。无论是哪种方式，它的本质都是想处理 **variable substitution**。有趣的是，「代数」的本质其实也是「代入」，但代数中的代入法则是 implicit 的，所以用代数来 explicitly 表达代入的概念反而很难。

至於  $\lambda$ -calculus，它本质上就是一种处理 substitution 的方案。而 **combinatory logic** 和  $\lambda$ -calculus 是等效的，后者消除了「变量」这概念，但取而代之的是式子的长度更复杂（比 relation algebra 更复杂）。

(First-order) predicate logic 的代数化，可以由 Alfred Tarski 的 **cylindric algebra** 给出：

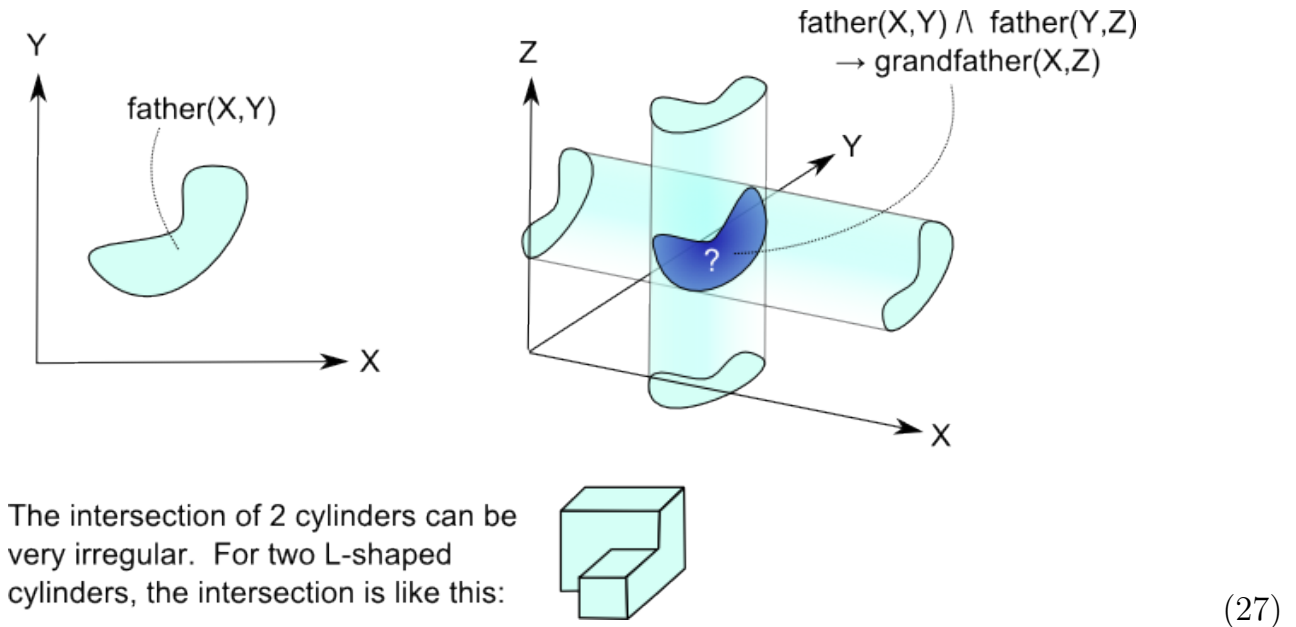
$$\frac{\text{propositional calculus}}{\text{Boolean algebra}} = \frac{\text{predicate calculus}}{\text{cylindric algebra}} \quad (25)$$

Higher-order logic (HOL) 的代数化可以用 **topoi theory** 给出 [6] [7]：

$$\frac{\text{intuitionistic propositional calculus}}{\text{Heyting algebra}} = \frac{\text{higher-order logic}}{\text{elementary topos}} \quad (26)$$

HOL 等效於 untyped  $\lambda$ -calculus，而 typed  $\lambda$ -calculus (= type theory) 较为弱一点。HOL 这个家族的特点，是“substitution of equal by equals”，这会令逻辑式子的长度增加。但 predicate logic 的做法是用**实物** (objects) 来做替换，它不会增加式子的长度；但将它代数化的结果是引入了 cylindrification、diagonal set 等概念，例如在 fig. (18) 有  $y_1 \equiv x_2$  这个 diagonal。

所谓 **cylindrification** 的意思是这样的：



$X, Y, Z$  代表 domains，即所有「人」或「物体」 (objects) 的集合。物体之间的**关系**是一些 regions（如蓝色那块平面代表“father”关系）。两个关系之间的“and”就是就是那两个 cylinders 的 intersection，所以叫 cylindric algebra。注意：这里的  $(X, Y, Z)$  不同於 fig. (18) 的  $(x_1, x_2, y_1)$ ，前者是 3 个 domain 上的 2 个关系的交集：

$$\text{father} \wedge \text{father} \quad (28)$$

后者是  $x_1 = \text{“father”}$  这个 predicate 的内部结构:

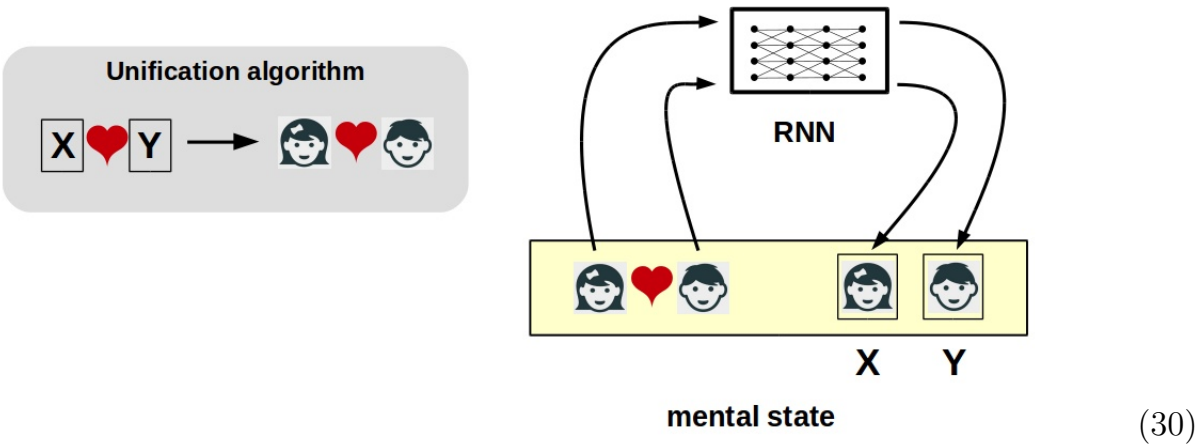
$$\text{father}(x_2, \cdot) \wedge \dots \mapsto \text{grandfather}(y_1, \cdot) \tag{29}$$

是个 diagonal (即 linkage:  $x_2 \equiv y_1$ )。

(这段的 references 太多，我有空再补上....)

2.1.8 用时间换取空间

上面说的是 substitution 在空间中的 linkage 结构。但也可以将 substitution 分拆成若干个简单的步骤。方法是：将某些内容放进记忆体中的「盒子」中，这些盒子充当 variables 的角色，也就是很具体地实现 substitution 的动作。换句话说，将空间复杂性转换成时间复性<sup>†</sup>；以下是示意图：



$X, Y$  是状态空间  $\mathbb{X} \ni x$  里面的「盒子」= variables。

先前我说的 linkage 结构，放在有限维向量空间比较方便，但它是 first-order logic，处理 higher-order 关系时有很大困难。Higher-order relations 似乎还是要用这种分拆步骤的方法解决。

2.1.9 Cartesian-closedness

折腾了这么久，但仍然缺少了一个重要的特性：Cartesian-closedness。它指的是在某个范畴内，对任意的  $A, B$ ，都必然可以找到它们的：

$$\boxed{\text{product}} \quad A \times B \quad \text{和} \quad B^A \quad \boxed{\text{exponentiation}} \tag{31}$$

在逻辑中这是指：

$$A \wedge B \quad \text{和} \quad A \rightarrow B \tag{32}$$

为什么需要 Cartesian-closed? 注意在 minimal architecture 里，只有两种记忆，即瞬时记忆  $x$  和永久记忆  $F = \boxed{\text{KB}} \vdash$ 。从 logic-based AI 的角度来看， $\boxed{\text{KB}}$  里装著 logic rules，但  $x$  里面

<sup>†</sup> 人脑的脑电波由 0.5 Hz 到 40 Hz 都有，我们感觉上瞬间的动作可能已经过了若干次迴路。

只有 ground facts。Ground sentence 指的是**没有变量**的命题，例如：

$$x = \text{见到地上有血迹} \quad (33)$$

相比之下，logic rule 是有变量的 conditional statement，例如：

$$\forall Z. Z \text{ 有血迹} \rightarrow Z \text{ 可能是凶案现场} \quad (34)$$

如果在  $x$  里面可以存放 rule，那表示  $x \ni \mathbb{X}$  是一个 Cartesian-closed category。这样的  $\mathbb{X}$  是一个更 powerful 的结构，亦即系统可以思考更复杂的 thoughts。更重要的是， $x \ni \mathbb{X}$  和  $F = \boxed{\text{KB}} \vdash$  现在**地位平等**，因为它们都是  $\mathbb{X} \rightarrow \mathbb{X}$  的**函数**，因为 Cartesian-closed 表示  $\mathbb{X} \simeq \mathbb{X}^{\mathbb{X}}$ 。这个特性在 belief revision 中似乎会很有用（见下节）。

如何在神经网络中做到 Cartesian-closed？记得神经网络中  $x$  是一组 features =  $(x_1, x_2, \dots, x_n)$ 。一个办法是将所有  $x$  都变成  $\mathbb{X} \rightarrow \mathbb{X}$  的函数。逻辑的 implication  $A \rightarrow B$  显然是函数，但单一 ground sentence  $A$  也可以是函数： $\top \rightarrow A$ ，其中  $\top$  是逻辑「真」。注意：这些函数中可以有 linkages，即处理变量的能力。

转到神经网络中， $A \rightarrow B$  是一个  $\mathbb{X} \rightarrow \mathbb{X}$  的神经网络。 $\top \rightarrow A$  也是一个  $\mathbb{X} \rightarrow \mathbb{X}$  的神经网络，但它将  $1 \mapsto A$ 。

那  $\mathbb{X}$  是一个怎样的空间？它本身可以是一个深度神经网络 weights，但这个神经网络的输入 / 输出层必须有足够的阔度去处理它自己！表面上似乎不可能.... 越多的 weights 需要更多的 weights 去处理.... 但如果令  $x$  储存一些 partial functions 或许可以。

### 2.1.10 应用在 belief revision

Belief revision（也可以叫“truth maintenance”）是经典逻辑 AI 发展的高峰。如果可以用我们的新 architecture 做到 belief revision，我们会很有信心这个理论是 general intelligence。

正常的逻辑运作模式是由  $\boxed{\text{KB}}$  作用在  $x$  上给出新的  $x'$ ：

$$\boxed{\text{KB}}(x) = x' \quad (35)$$

但 belief revision 或者可以看成是  $x$  作用在  $\boxed{\text{KB}}$  上的结果：

$$x(\boxed{\text{KB}}) = \boxed{\text{KB}}' \quad (36)$$

由於 Cartesian-closedness， $x$  和  $\boxed{\text{KB}}$  的地位是平等的，使上面的运作成为可能。

这只是一个 vague idea，我会再回到这里填补这空白....

## 2.2 命题-level 结构

### 2.2.1 思维状态分拆成命题

将思维状态  $x \in \mathbb{X}$ （你当下所思想的东西）分拆为命题的集合，是有必要的。 $x$  由一些 **thoughts**（思维单元）组成，一个 thought 对应於逻辑中的一条**命题**，举例来说：

$$x_1 = \text{我正在上课} \wedge \text{我很肚饿} \wedge \dots \quad (37)$$

也可以有另一个状态：

$$\mathbf{x}_2 = \text{我正在搭地铁} \wedge \text{我很肚饿} \wedge \dots \tag{38}$$

如果不分拆的话， $\mathbf{x}$  和  $\mathbf{x}_2$  会是两个完全不同的状态。分拆之后，它们有共同的因子，这些因子可以 factor out 来处理，换句话说可以用较小的  $\boxed{\text{KB}}$  处理大量的情况，达到资讯压缩的 economy。

### 2.2.2 Boolean algebra

A **Boolean algebra** is a structure with:

$$\begin{array}{ccc} \text{underlying set} & \text{binary ops} & \text{unary op} \\ & \wedge, \vee, \neg & \\ \mathcal{B} = (A, & & \end{array} \tag{39}$$

但似乎最重要的结构（尤其是当引入了 fuzzy-probabilistic 之后）是这个 commutativity:

$$\forall a, b \in \mathcal{B}. \quad a \wedge b = b \wedge a \tag{40}$$

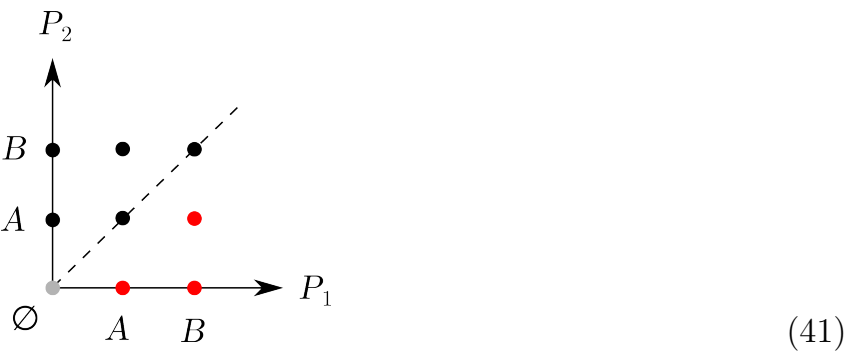
更简单地讲：状态  $\mathbf{x}$  只需要是一个命题的集合，命题的次序和相同重复可忽略。

如果放到 vector space 上，这结构是一个 **symmetric algebra**，亦即一个 tensor algebra 加上 commutative 的条件。（但我暂时没有用这方面的理论）

### 2.2.3 命题 → 向量空间

目的是想将所有命题 map 到向量空间上，次序和重复不重要。

考虑最简单的情况： $P_1$  和  $P_2$  是两个命题的容器，思维状态 =  $(P_1, P_2)$ ，而每个命题可以是  $A$  或  $B$ 。我们可以这样摆放  $P_1 \times P_2$  空间的元素：



例如红色那 3 点分别代表  $\{A\}, \{B\}, \{A, B\}$ 。对角线上的元素可以不要，因为像  $(A, A)$  等是多馀的重复。换句话说，全体命题的集合是  $\{\text{lower-triangular 那些元素}\} \setminus \text{diagonal} \cup \emptyset$ 。

在高维情况下这种空间的节省很有效率：symmetrize 之后，hypercube 的一角的体积只有原体积的  $1/n!$ ，这是很好的结果，因为维数灾难以指数式增加，但  $n!$  完全足够抵销它。

然后考虑这对称空间上面的函数，它们的结构会怎样改变？未 symmetrize 之前的函数定义域是：

$$\mathbf{F} : \mathbb{X} \rightarrow \mathbb{X} \quad = \quad \mathbb{P}^n \rightarrow \mathbb{P}^n \tag{42}$$

其中  $\mathbb{P}$  是单个 proposition 的空间。Symmetrize 之后变成：

$$\mathbf{F} : \text{sym}(\mathbb{P}^n) \rightarrow \text{sym}(\mathbb{P}^n) \quad (43)$$

后者的空间  $\text{sym}(\mathbb{P}_1, \mathbb{P}_2, \dots, \mathbb{P}_n)$  有某个 **order**。换句话说，如果输入是  $(p_1, p_2, \dots, p_n)$ ，需要按照  $\mathbb{P}$  上的 order 对  $p_i$  排序，然后才呼叫  $\mathbf{F}$  计算。

在  $\mathbb{P}$  上的排序是很简单的，因为每个  $p \in \mathbb{P}$  已经有其位置（位置是深度学习 learn 出来的）， $p_1 > p_2$  可以由向量空间的 cone 定义，然后在  $\mathbb{P}^n$  上用 lexicographic ordering 即可。

在这节提出的算法很简单，但可以带来  $1/n!$  的改进，很值得试验它的实际效果 ☺

### 3 结论

本篇分析了逻辑的基本结构，然后将这些结构附加到神经网络上，以加速学习。现正计划写代码验证这些想法的有效性。

## Acknowledgement

谢谢 Ben Goertzel（OpenCog 人工智能的创始人）在 AGI mailing list 上和我的讨论。Ben 初次指出神经网络学习和逻辑 inductive 学习的不同，引起我研究两者之间的关系。

## References

1. Wikipedia: Deep dreaming (<https://en.wikipedia.org/wiki/Deepdreaming>).
2. Wikipedia: Universal approximation theorem ([https://en.wikipedia.org/wiki/Universal\\_approximation\\_theorem](https://en.wikipedia.org/wiki/Universal_approximation_theorem)).
3. Robert Gilmore and Marc Lefranc. *The topology of chaos: Alice in stretch and squeezeland*. Wiley-VCH, 2011.
4. Hobbs. interpretation as abduction, in book "discourse and inference", Nov 2003.
5. Jerry R Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 95–103. Association for Computational Linguistics, 1988.
6. Joachim Lambek. Categorical versus algebraic logic. In Andr  ka, Monk, and N  meti, editors, *Algebraic logic*, pages 351–360. North-Holland, 1988.
7. Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic – a first introduction to topos theory*. Springer, 1992.
8. Roger Maddux. *Relation algebras*. Elsevier, 2006.
9. Mikolov, Sutskever, Chen, Corrado, and Dean. Efficient estimation of word representations in vector space. *Proceedings of workshop at ICLR*, 2013.
10. Luc De Raedt. *Logical and relational learning*. Springer, 2008.
11. Gunther Schmidt. *Relational mathematics*. Cambridge, 2010.



12. Stephen Smale. Differentiable dynamical systems. *Bulletin of the American Mathematical Society*, 1967.
13. Tamás Tél and Márton Gruiz. *Chaotic dynamics: an Introduction based on classical mechanics*. Cambridge, 2006.
14. King Yin Yan. ILP tutorial  
[https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/genifer/Genifer-induction\(30July2012\).pdf](https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/genifer/Genifer-induction(30July2012).pdf).
15. King Yin Yan. The structure of memory. (to be submitted AGI-2017).
16. King Yin Yan, Juan Carlos Kuri Pinto, and Ben Goertzel. Wandering in the labyrinth of thinking – a cognitive architecture combining reinforcement learning and deep learning. (to be submitted AGI-2017).